

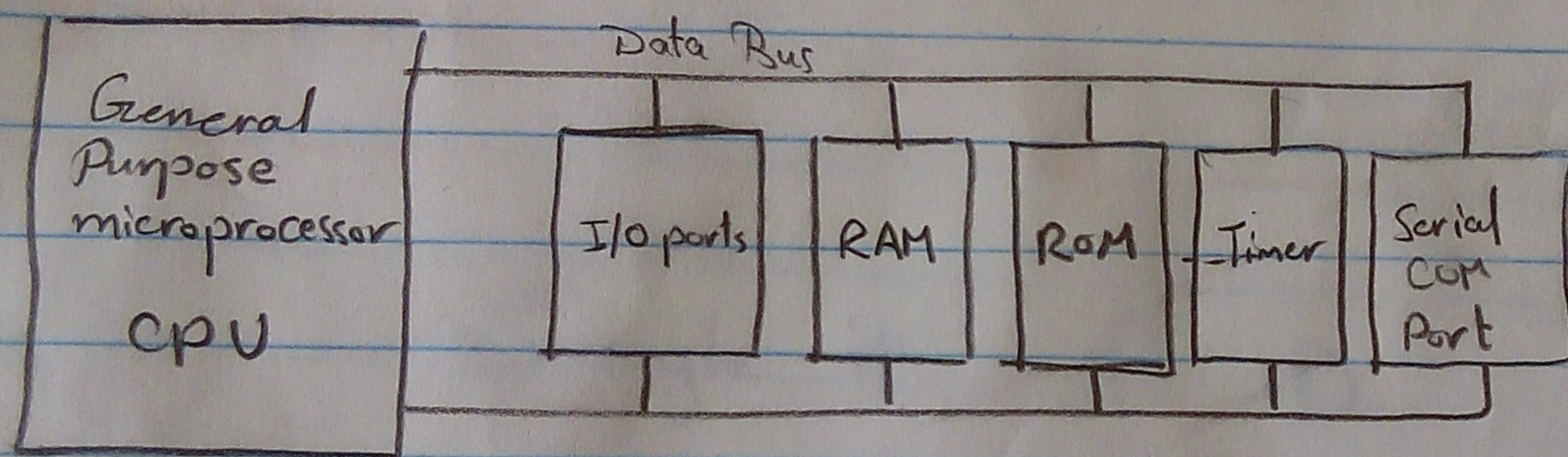
Microcontroller

Sheet 1

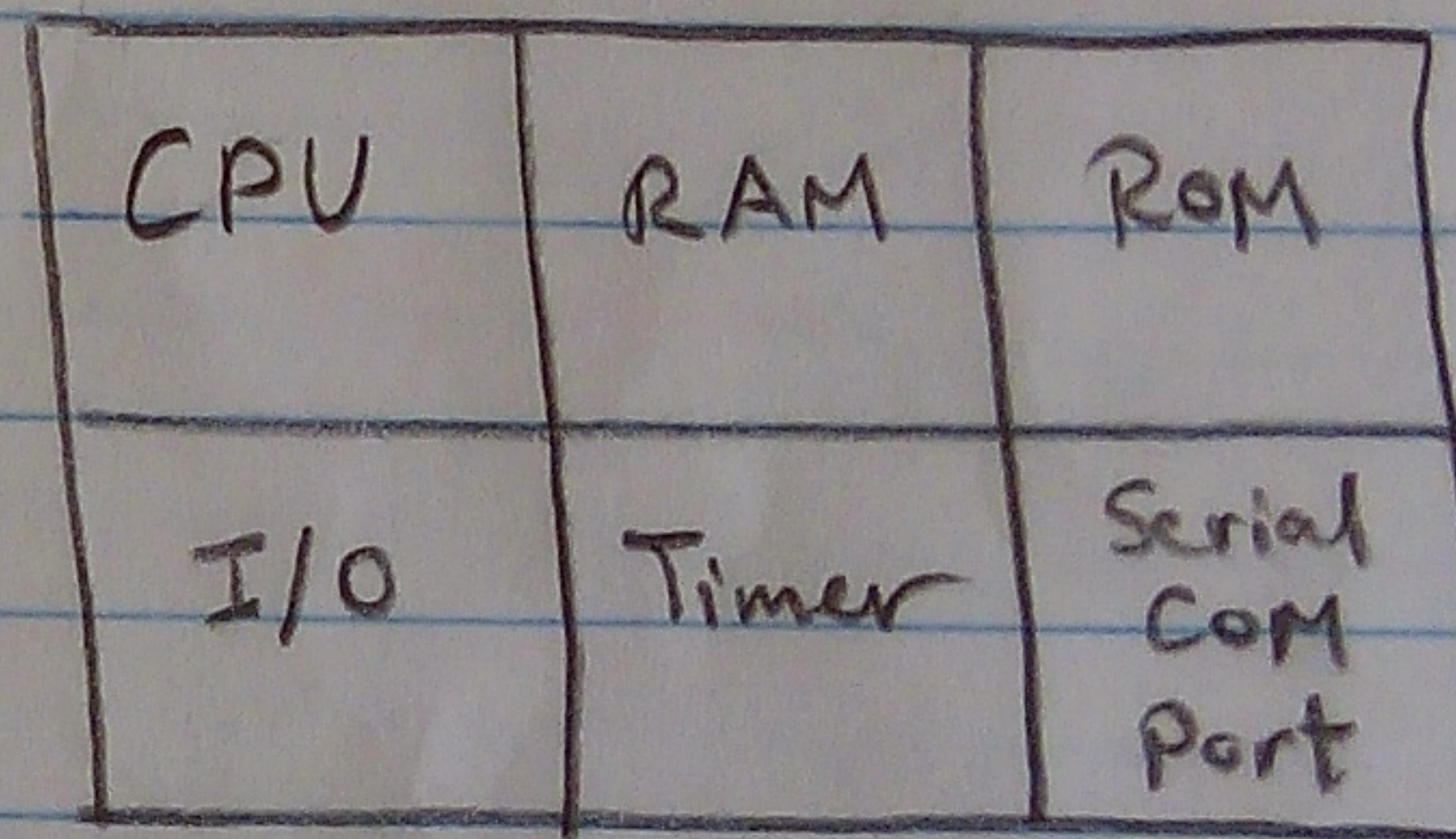
Q1: What is the difference between microcontroller and microprocessor

Microprocessor is the Central Processing Unit (CPU) of any Computer System where all logical and mathematical operations are performed. μP has no RAM, no ROM, no I/O ports. It must be connected to other peripherals to perform the required task (i.e. to form the Computer System).

Microcontroller is a Computer System on a single chip. MC consists of a microprocessor, RAM, ROM, I/O ports and other peripherals. All these components are embedded in the same IC.



Computer System with microprocessor



Microcontroller

Q2: Why do microcontrollers exist at all? Why not just use a normal processor and add all necessary peripherals externally?

The necessity of adding peripherals to microprocessor based system makes the system bulkier and much more expensive. (i.e. The cost of IC fabrication will increase if we increase the number of ICs in the system).

The presence of more ICs makes the space required for the system larger and the power consumption will increase.

In many applications the space the hardware takes, ~~and~~ power consumption and price per unit are much ^{more} critical considerations than computing power. Hence the microcontroller will be much more suitable.

Q3: What do you believe are the three biggest fields of applications for microcontrollers? Discuss your answer with other students

Search for the answer and discuss it with your colleagues

OR

:D :)

Go to Slide 46 in the reference :

"The Text Book of 8051 microcontroller and embedded Systems"

Q4: Visit the homepage of some electronics vendors and compare their stock of microcontrollers.

a) _____

b) _____

c) _____

Google it :P :D

Q5: Name the Basic Components of a microcontroller. For each component, give an example where it would be useful.

Microcontroller Components are :

1. CPU : In which all operations are performed (Instructions execution)
2. ROM : The memory where we store the program that perform the task
3. RAM : The temporary memory required when executing programs
4. I/O ports : The hardware interface between microcontroller CPU and the external world
5. Timers : An important part of any microcontroller is to perform an action within fixed time calculated by Timers of MC
6. Serial COM port : For Serial Communication (Data transfer) between MC and other devices in the system

CPU	Timers
RAM	ROM
I/O	Serial Port

Q6: What is an embedded System? what is a real time System? are these terms Synonyms? Is one a subset of the other? why or why not?

Embedded System is a Computer System with a dedicated function within larger electrical or mechanical system. Embedded Systems are always designed to control the operation and performance of the original system.

The Core of any Embedded System is the microcontroller, we can call the microcontroller the brain of the system. (Ex: Washing-Machine Controller)

Real-time System is a system in which time consideration is very critical. The response of the system must occur within fixed time constraint. If the results reached after the time the result are considered wrong (i.e. System failure). (Ex: Air-Bag Controller of modern Cars)

Real-time Systems are Subset of Embedded Systems. (i.e. they are embedded systems whose performance must be as faster as possible to meet system deadlines).

Q7: Why are there so many microcontrollers? wouldn't it be easier for both manufacturers and consumers to have just a few types?

Microcontrollers can be used for controlling different systems with different size, price and accuracy. According to system needs we choose microcontroller.

The variety of manufactures in the market make them compete with each other. This affects positively in the quality of products.

Generally there are criteria for choosing microcontrollers. These criteria are listed in slide 50 of the reference.

"The text Book of 8051 microcontroller and embedded Systems" - Mazidi

Micro controller

Sheet 2

Q1: Explain what will happen if the following codes are executed?

- 1- READ: MOV A, R1 ; move the content of R1 to A
ANL A, #2H ; Anding of content of A and 2H
CJNE A, #02H, READ ; Compare content of A and 02H, if Not equal Jump to instruction labeled by READ
MOV R3, #0FFH
- 2- STAT: MOV A, #01H ; move the content of H to A
JNZ STAT ; if A not equal zero Jump to STAT

here A always is not equal zero (Infinite loop)

Q2: Find the number of bytes for each of the following instructions:

- | | |
|----------------|---|
| 1- MOV A, #45 | 2 bytes (1: opcode + 1: data) |
| 2- INC R1 | 1 byte (1: opcode) |
| 3- MOV R1, #30 | 2 bytes (1: opcode + 1: data) |
| 4- MOV R3, A | 1 byte (1: opcode) |
| 5- JNZ Label | 2 bytes (1: opcode + 1: address) |
| 6- ADD A, R1 | 1 byte (1: opcode) ↑
Short Jump |
| 7- ADDC A, 40 | 2 bytes (1: opcode + 1: address) |
| 8- ANL A, #30 | 2 bytes (1: opcode + 1: data) ↑
1-byte memory address |

Q3: Write an assembly program that finds either product or sum of two numbers NUM1, NUM2 as follows:

if $NUM1 < NUM2$, Find the product

if $NUM2 < NUM1$, Find the summation

ORG 0000

MOV A, #NUM1

MOV R1, #NUM2

SUB A, R1

JC product ; Jump if $NUM1 < NUM2$

MOV A, #NUM1

ADD A, R1

SJMP finish

Product: MOV A, #0H

again: ADD A, #NUM1

DJNZ R1, again ; Multiplication is repetitive addition

finish: END

Q4: Write an assembly Program that Stores NUM1 Continuously in 30 odd memory locations starting from the address 10 H

ORG 0

MOV A, #NUM1

MOV R1, #10H ; The address of memory location stored in R1

MOV R2, #30 ; 30 (decimal) the number of locations

again: MOV @R1, A

ADD R1, #02

DJNZ R2, again

END

Q5: Write an assembly program that multiplies the number in the memory location DATA1 with itself 30 times and stores the result in the memory location DATA2

Assuming the number is n then the required is to compute : $(n)^{30} = \underbrace{n \times n \times \dots \times n}_{30 \text{ times}}$

every multiplication can be implemented as n -addition operations

ORG 0

MOV R1, #29

MOV R3, DATA1 ; MOV R3, DATA1

Multiply: MOV R2, DATA1

MOV A, #0 ; MOV A, #0

ADDITION: ADD A, R3 ; ADD A, R3

DJNZ R2, ADDITION

MOV R3, A

DJNZ R1, Multiply

END

$n \times n : n\text{-additions}$

$\downarrow$
 $a \times n : n\text{-addition of } a$

$\downarrow$
 $b \times n : n\text{-addition of } b$

$\vdots$

Repeat this process

29 times

Q62 Write an assembly program to look for a number NUM in 300 successive memory locations starting from 00H. The accumulator is either loaded with the number (if it exists) or zero

ORG 00H

~~MOV R3, #00H~~

MOV A, #NUM

MOV DPTR, #00H

MOV DPL, #00H

MOV R1, #30

Outer: MOV R2, #10

Inner: CJNE A, @DPTR Not-Equal

SJMP Finish ← MOV R3, A

Not-Equal: INC DPTR

DJNZ R2, Inner

DJNZ R1, Outer

Finish: MOV A, R3

END

Note: 300 locations

Can't be

Counted by

one loop

(max = 255)

Solⁿ: We use two

nested loops

Note: For addresses

greater than

255 we need

two bytes to

represent the

address

I am not Sure

That this Code will
run on a real Hardware

:D

Solⁿ: Use DPTR

register

DPA: High Address

DPL: Low Address

Q7: Write assembly program that stores (in ascending order) N numbers stored in memory starting from the address aaH

ORG 0

MOV R1, #1

Outer: MOV R2, #N ; number of elements

SUB R2, R1

MOV R3, #aa

Inner: MOV R4, R3

INC R4

MOV A, @R3

SUB A, @R4

JC OK

~~CALL~~ SWAPR

OK: INC R3

DJNZ R2, Inner

INC R1

; i used in calculation of n-i

SUB R2, R1

JNC Outer

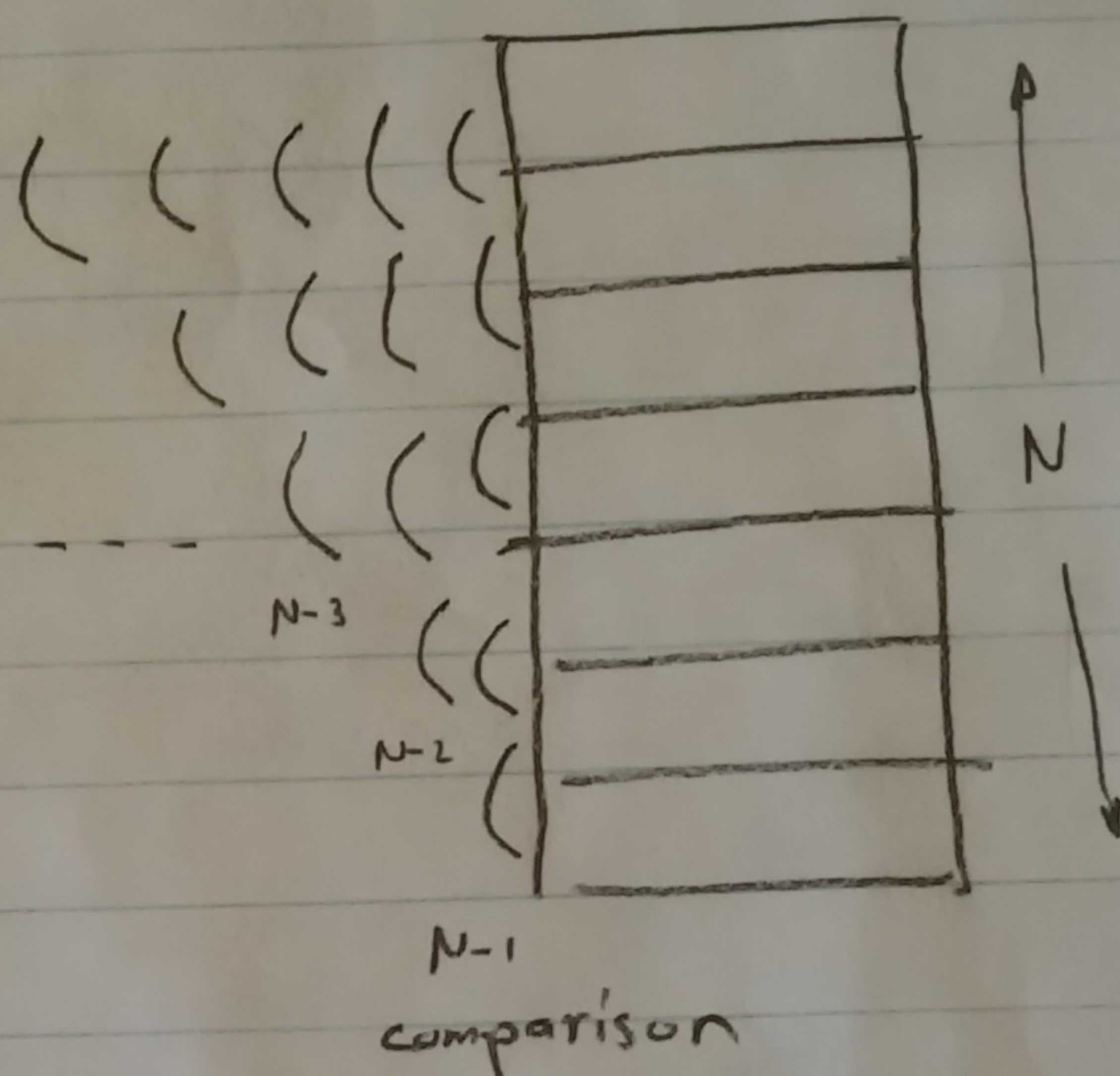
END

SWAPR: MOV A, @R4

MOV @R4, @R3

MOV @R3, A

RET



Micro Controller

Sheet 3

Q1: Find the number of times of the following loop is performed

```
MOV R3, #150      ; initialize R3 with 150
LLL: MOV R4, #180  ; initialize R4 with 180
HHH: DJNZ R4, HHH  ; R4 is decremented to zero (180 iteration)
      DJNZ R3, LLL  ; For each time R4 reaches zero, R3 is
                    ; decremented and R4 initialized again with
```

$$\begin{aligned} \text{number of times} &= \text{Innerloop} \times \text{outer loop} \\ &= 180 \times 150 = 27000 \end{aligned}$$

Q2: Show the Code to execute an action 1000 times

Note 1000 > 255 Can't be added to one register
Hence, we use nested loops.

Let Inner Loop repeats the action 100 times

Let outer Loop repeats the Inner loop 10 times

Then we repeat the action 1000 times

Code:

```
MOV R1, #10
Outer: MOV R2, #100
Inner: DJNZ R2, Inner
      DJNZ R1, Outer
      END
```

General form:

```
MOV R1, #10
Outer: MOV R2, #100
Inner: *Some Code*
      DJNZ R2, Inner
      DJNZ R1, Outer
      END
```

This Code is repeated 1000 times

Q3: Write an assembly program to add values in registers R2, R3, and R5. Complement the result and then store the lower byte in accumulator and the higher byte in R6.

```
MOV R6, #0
MOV A, R2
ADD A, R3           ; Add Content of R3 to Content of A (R2)
JNC Low1           ; if no Carry occurs then no need for High byte
MOV R6, #01        ; if Carry occurs we add one to High byte
Low1: ADD A, R5
JNC Low2           ; if no Carry occurs then no need for High byte
ADD R6, #01        ; if Carry occurs we add another one to R6
Low2: CPL A        ; Complement Low byte
MOV R4, A          ; Move low byte to temporal location
MOV A, R6          ; Move High byte to A
CPL A              ; Complement High byte
MOV R6, A          ; Restore High byte in R6
MOV A, R4          ; Restore Low byte in A
END
```

Q4:

Q4: Write an assembly Program to add the contents of register R3 to the accumulator 300 times and store the result in memory location LOC.

Note: 300 times will be implemented using nested loops (30x10)

```
MOV R2, LOC
MOV R4, #30
Outer: MOV R5, #10
Inner: ADD A, R3
       DJNZ R5, Inner
       DJNZ R4, Outer
MOV @R2, A      ; indirect addressing
END
```

Q5: What would happen if the program didn't initialize the SP? Could you overwrite Code? Could you overwrite data?

Note: The default value of SP is to point at Register 07 of the memory. The default section in RAM reserved for stack is Bank 1.

* If we don't initialize SP it will be (by default) initialized by 07 (the last location in bank 0). PUSH instruction will ① increase SP

② add data/address to the new location (i.e. SP always points to top of Stack)

* If we don't initialize SP we may overwrite data stored in memory locations from 08 (Stack is a section of RAM). No Code will be affected (Stack doesn't belong to ROM).

Q6: Let us assume that you have initially set the SP to the end of the RAM. If you push the contents of R0 on the stack after it has been initialized, then at which address is the data located and to which address does the SP point?

Note: Push instruction is executed as follows

- 1- the SP is incremented by one
- 2- data stored in the Current (after increment) address of SP

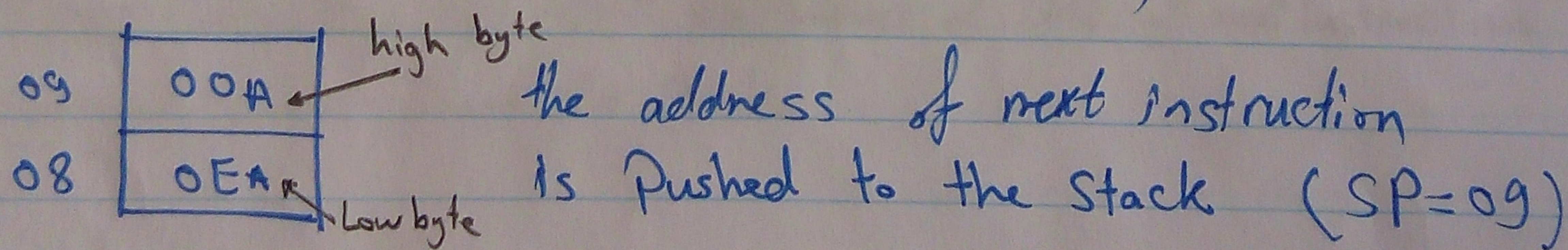
when $SP = FFH$ (The end of memory)

if the instruction $PUSH\ 0$ is executed

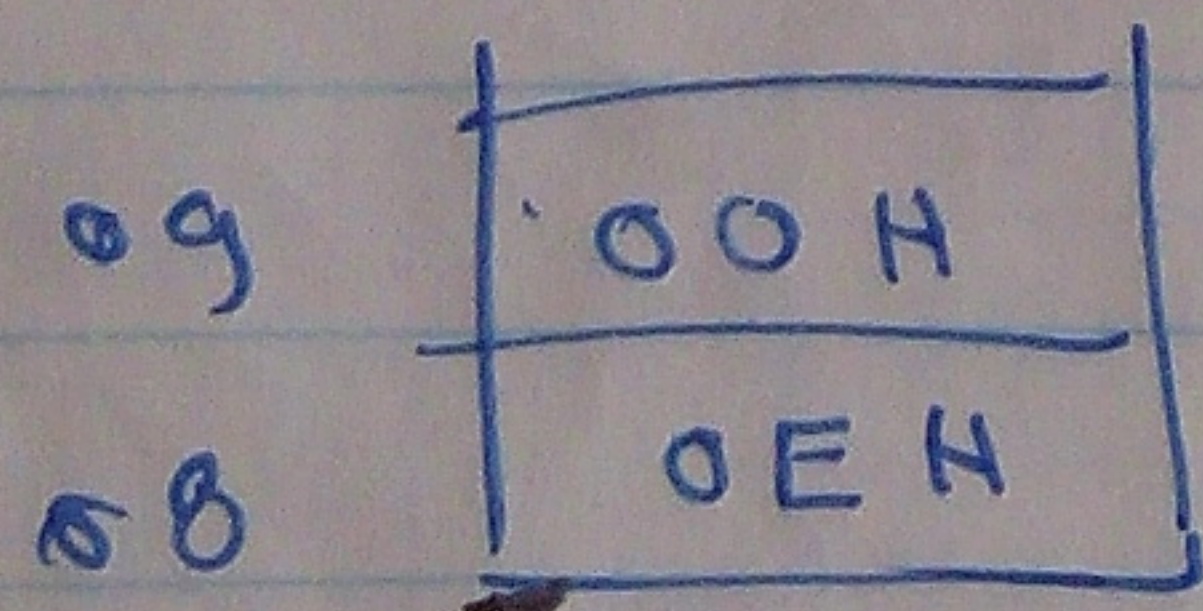
- ① SP incremented by one $SP = 00$
- ② data will not be transferred to memory location 00
Stack overflow

Q7: Show the stack for the Code (Look at Q7 sheet 3)

when executing instruction (LCALL DELAY)



when executing instruction (RET)



the data (address of next instruction) is Popped from stack and is Put in the accumulator
($SP=07$)

Note: The data hasn't been deleted but the SP is changed to point to another location

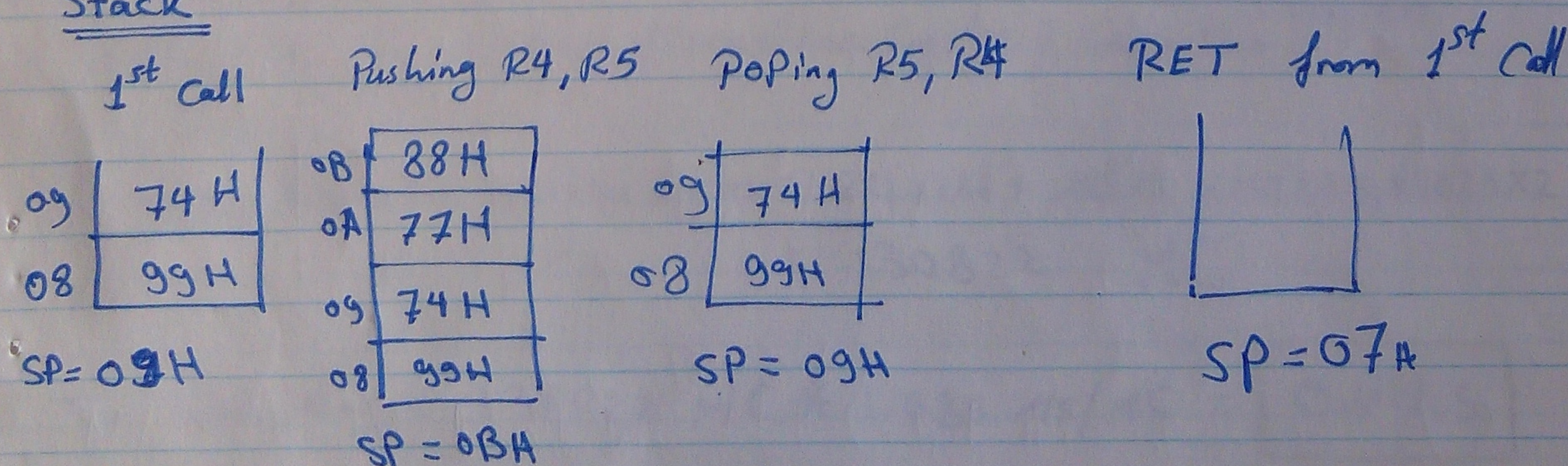
Q8: Explain what will happen if the following code is executed showing the Stack. Calculate the time taken by the DELAY Subroutine

- 1- The program initialize Accumulator by 66H
- 2- The value of A is used as Output for port P1
- 3- Move the value 77H to R4 and the value 88H to R5
- 4- Call delay Subroutine to get the required time delay
- 5- After return from delay Subroutine, the program move the value 99H to the accumulator
- 6- The value of A is used as Output for port P1
- 7- Call the delay Subroutine again
- 8- Repeat the above Sequence

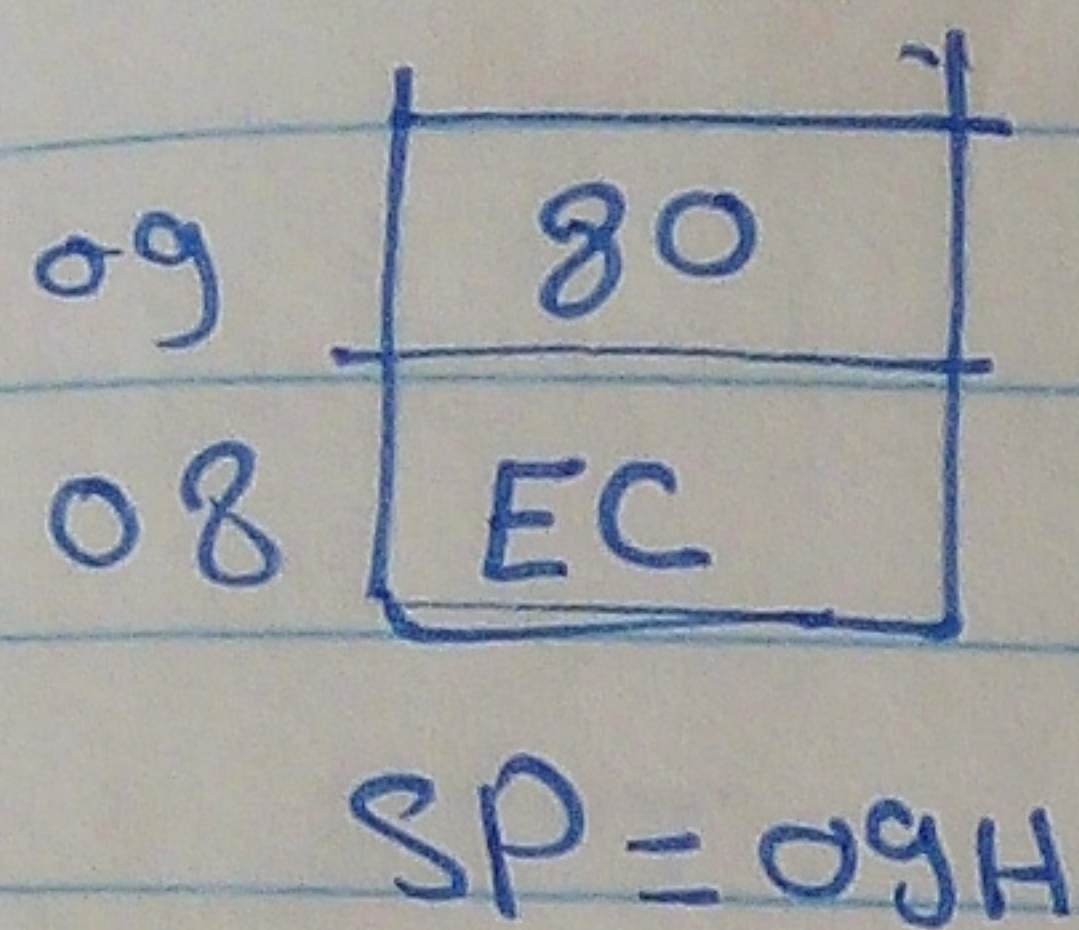
Delay Subroutine

- 1- Calling a Subroutine (LCALL) will push the address in PC to the Stack
- 2- In Subroutine, the program here push the content of R4 and R5 to the Stack
- 3- The delay Subroutine, then, Perform a nested loop (255 x 255) to make delay
- 4- Before return from Subroutine the original values of R4 and R5 are restored from the Stack.
- 5- RET instruction will POP the values of PC of next instruction in the original Code from the Stack

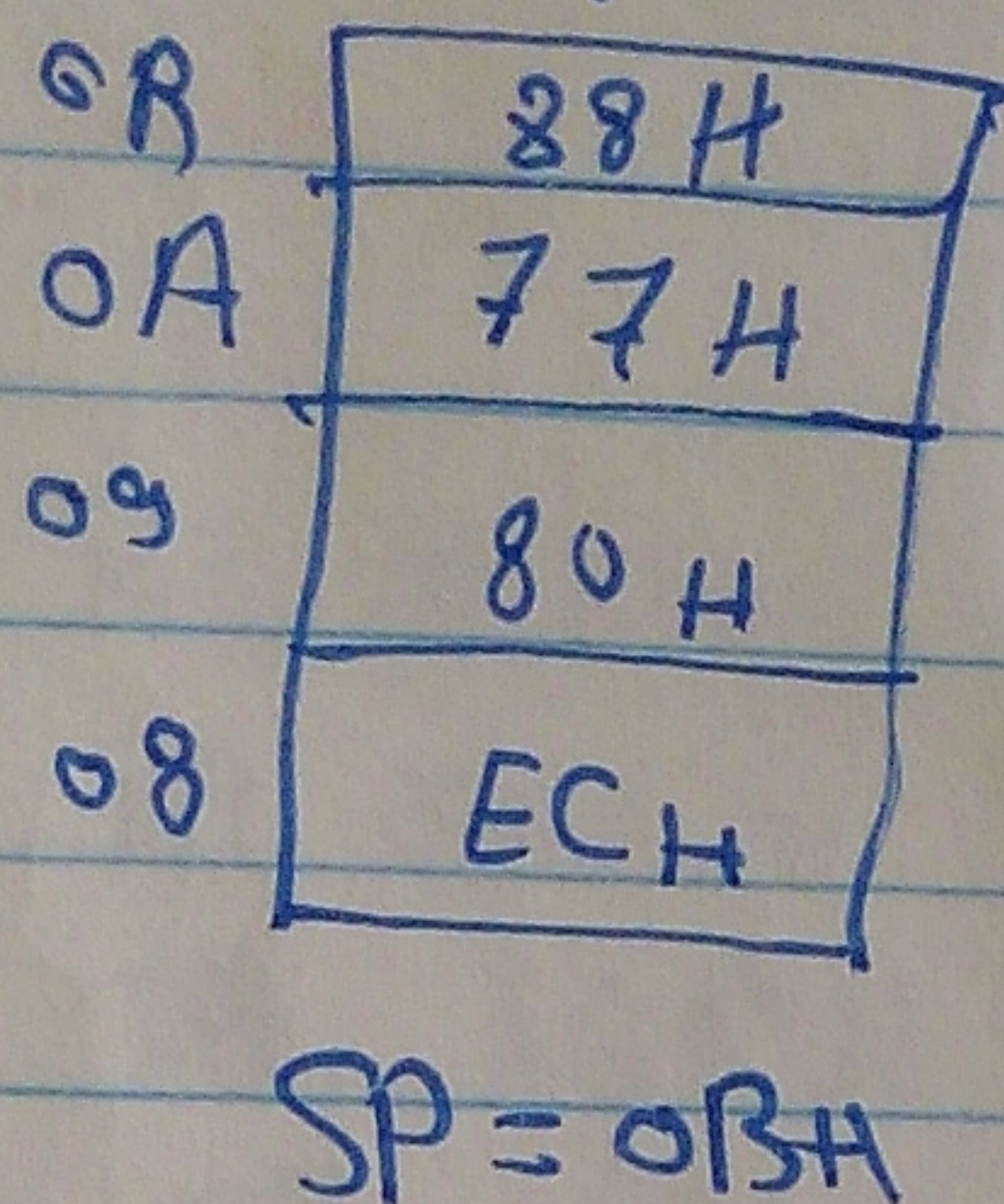
Stack



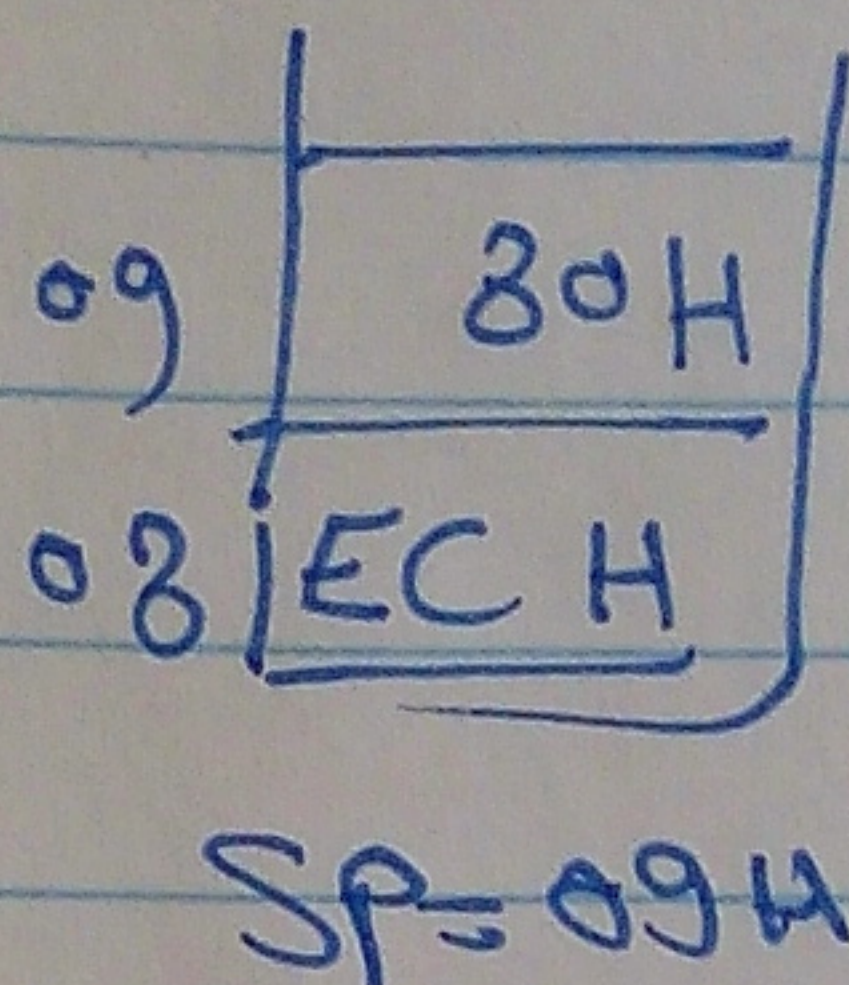
2nd Call



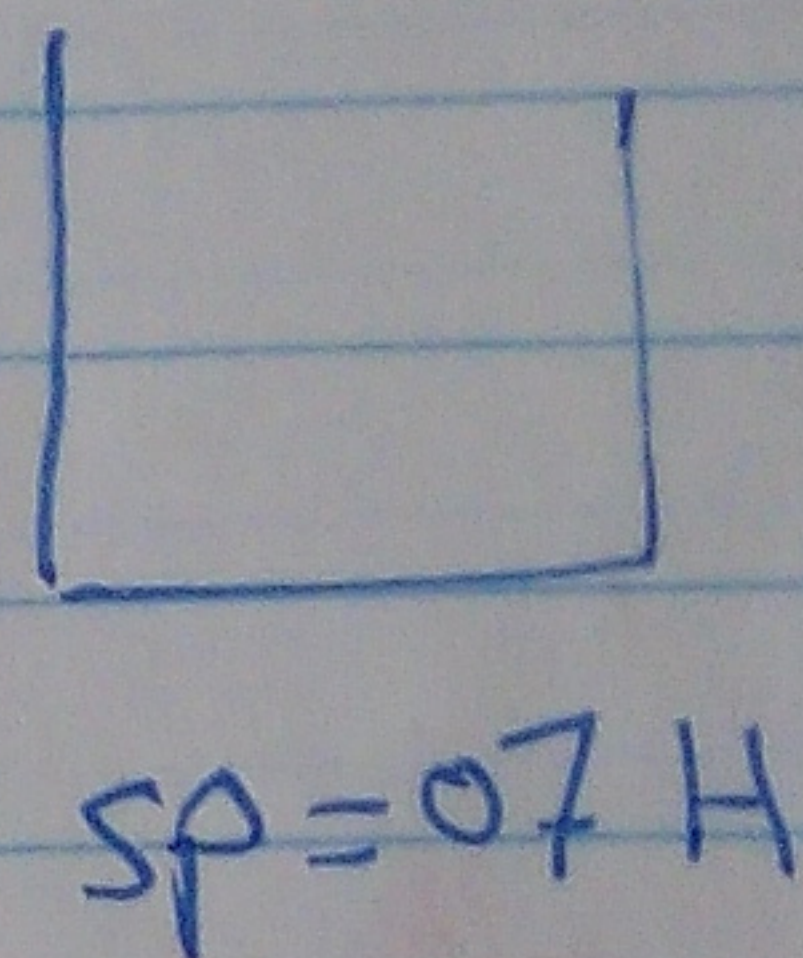
PUSH R4, R5 to stack



POP R5, R4



RET from Subroutine



Delay:

To Calculate the delay we calculate the time required to execute the Subroutine.

number of instructions in Subroutine

	Machine Cycles
2 PUSH Instructions	1
1 MOV instruction	1
255 MOV instruction (Outer loop)	1
255 Decrement and Compare (outer loop)	2
255 X 255 Decrement and Compare (inner loop)	2
2 POP Instructions	1
1 RET instructions	2

For 8051 assume we use Crystal of $f_s = 11.0592 \text{ MHz}$

Each instruction requires 12 clocks (MC requires $\frac{12}{11.0592 \text{ MHz}} = 1.085 \mu\text{s}$)
multiplied by number of machine Cycles / instruction

$$\begin{aligned} \text{The delay Subroutine machine Cycles} &= 2 \times 1 + 1 \times 1 + 255 \times 1 + 255 \times 2 + 25025 \times 2 + 2 \times 1 + 1 \times 2 \\ &= 130822 \text{ MC} \end{aligned}$$

$$\text{The total delay} = 130822 \text{ MC} \times 1.085 \mu\text{s/MC} = \boxed{0.142 \text{ s}}$$

For 8051 System of 11.0592 MHz, find how long it takes to execute the following code?

```
MOV R3, #200
NEXT: MOV R4, #0FFH
AGAIN: DJNZ R4, AGAIN
      DJNZ R3, NEXT
```

This is the correct code, there is a mistake in the code of the sheet

For 8051 each machine cycle requires 12 clock

$$\therefore \text{The time for one machine cycle} = \frac{12}{11.0592 \times 10^6} = 1.085 \mu\text{s}$$

Crystal Oscillator frequency

Each instruction requires certain number of machine cycles

To Calculate the delay

- ① Calculate the number of repetitions for each instruction
- ② The machine cycles of the instruction (from datasheet)
- ③ The total number of machine cycles $\times$ the time of one machine cycle

The above code

Instructions	Machine Cycles
1 MOV instruction	1
200 MOV instruction	1
200 $\times$ 255 Decrement & Compare	2
200 Decrement & Compare	2

$$\text{Total MC} = 1 \times 1 + 200 \times 1 + 200 \times 255 \times 2 + 200 \times 2 = 102601$$

$$\text{Total time} = 102601 \times 1.085 \mu\text{s} = \boxed{0.11 \text{ s}}$$